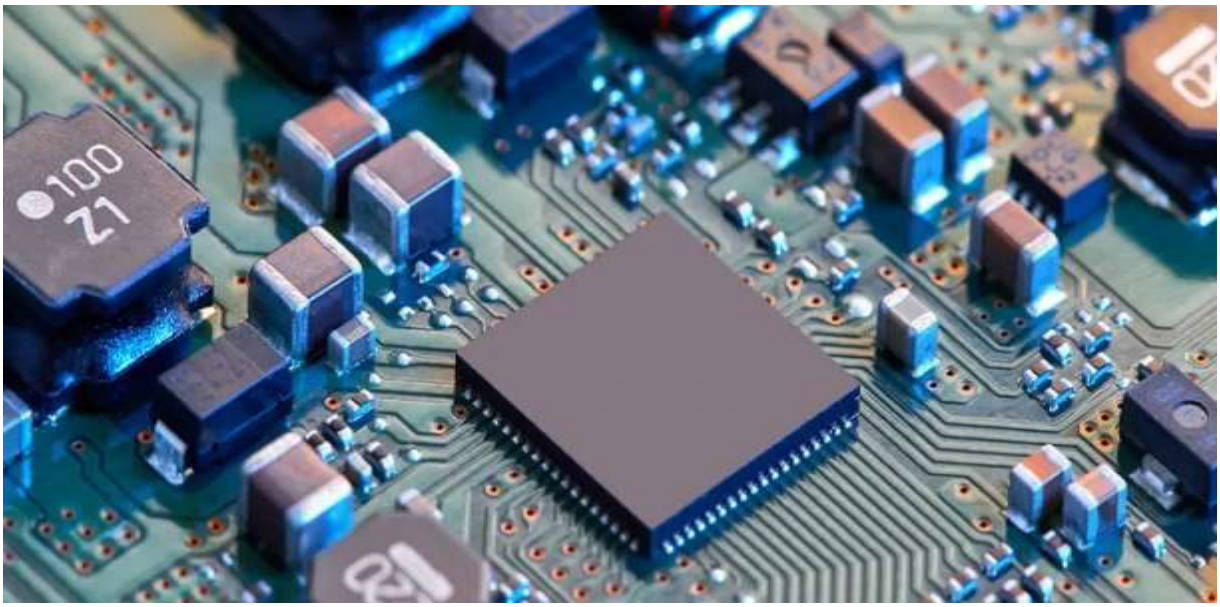
Proyecto final -Fase 2



Equipo 2 D14

1. Oscar Beltran Villegas | **Scripter**.
2. Mariscal Viorato Maria Fernanda | **Documenter**.
3. Ramírez Vázquez Ezequiel Alberto | **H-Designer**.
4. De La Cruz Chavarría Christian Israel | **Admi**.

Introducción

- Implantación de todas las instrucciones de tipo I que están predispuestas en esta nueva fase.
- Se generarán distintos buffers cada uno con sus respectivas entradas y salidas con un alcance de 32 bits, además de manejar todos, una entrada con un ciclo de reloj.
- En estos buffers se formarán bloques always y cada uno tendrá un final consecuente con el siguiente modulo.

Instrucciones de tipo I

Instrucción	Tipo	Sintaxis	Definición
Addi	I	Addi $rt,$rs,10	Suma inmediata
Subi	I	Subi $rt,$rs,10	Resta inmediata
Ori	I	Ori $rt,$rs,10	Instrucción lógica Or inmediata
Andi	I	Andi $rt,$rs,10	Instrucción lógica And inmediata
Lw	I	Lw rt,offset	Cargar en la memoria como valor con signo
Sw	I	Sw rt,offset	Almacena en la memoria
Slti	I	Slti $rt,$rs,10	Comparación si el resultado es menor que inmediata
Beq	I	Beq $rs,rt,offset	Para comparar en el banco de registros, haga una rama condicional relativa a PC y para registrar el resultado de una comparación menor que
Bne	I	Bne $rs,rt,offset	Para comparar el banco de registros y haga una rama condicional relativa a PC
J	I	J target	Para bifurcar en la región actual alineada de 256MB
Bgtz	I	Bgtz $rs,offset	Para probar en un banco de registros y haga una rama condicional relativa a PC

Desarrollo de nuevos elementos /buffers

IF_ID

El Instruction fetch ID (IF_ID) es un buffer. Este esta conformado por tres entradas, dos de ellas de 32 bits, y la tercera de un bit (esta es del ciclo del reloj) y dos salidas también de 32 bits estas son de tipo registro. Este Buffer está conformado por un bloque always, el cual sirve para realizar cambios cuando el ciclo del reloj se encuentre en positivo. Lo que se ejecuta en este modulo es que nuestras salidas van a ser igual a las entradas que vengan del siguiente modulo.

WB

El WB esta conformado por dos entradas, una que de las entradas es de dos bist y la otra de un bit que es del ciclo del reloj y una salida de dos bits que es de tipo registro. Este WB esta conformado por un bloque always donde nos arroja que la salida va a ser igual a la entrada del siguiente modulo.

M

Este modulo es un vector que esta conformado por dos entradas, una de estas entradas es de tres bits la otra entrada que nuevamente es de un bit que es del ciclo del reloj y una salida de tres bits que es de tipo registro. Este vector se conforma por un bloque que es de tipo always que nos arroja que la salida va a ser igual a la entrada del siguiente modulo .

EX

El modulo EX es un vector que esta conformado por dos entradas. La primera entrada es de cinco bits y la segunda entrada es de un bit que es del ciclo del rejoj y una salida de cinco bits que es de tipo registro. Este vector se conforma por un bloque always que genera que la salida sea igual a la entrada del siguiente modulo.

ID_EX

El ID_EX es un buffer que esta conformado por un total de siete entradas. Cuatro de sus entradas están conformadas de 32 bits, otras dos entradas de 5 bits, y una entrada de un bit que es del ciclo del reloj y un total de seis salidas que son 4 de ellas de tipo registro de 32 bits y dos salidas mas también de tipo registro de 5 bits.

Este buffer esta conformado por un bloque always que nos dice que cuando el ciclo de reloj sea positivo este hará cambios, además de que esta forma genera que sus salidas sean iguales a las entradas de los siguientes módulos.

EX_MEM

El EX_MEM es un buffer que esta conformado por tres entradas de 32 bits, una entrada de 5 bits y otras dos entradas de un bit (estas están conformadas por el z_flag y el ciclo del reloj) esta conformado por un total de 5 salidas, las primeras tres son de 32 bits, otra salida de 5 bits y la ultima salida de 1 bit, la cual es la del z_flag.

El buffer EX_MEM esta conformado por un bloque always, el cual sirve para que cuando el ciclo de reloj sea positivo, este pueda hacer cambios, además de que esta forma que las salidas de este modulo sean iguales a las entradas del siguiente modulo.

MEM_WB

El MEM_WB es un buffer que tiene un total de cuatro entradas, de las cuales dos entradas son de 32 bits, otra de las entradas es de 5 bits y la ultima entrada es de 1 bit y es la del ciclo de reloj. Además de contar con dos salidas de tipo registro de 32 bist y una salida mas tambien de tipo registro de 5 bits.

Este buffer está conformado por un bloque de tipo always que nos dice que cuando el ciclo de reloj este en positivo, este realice cambios y genera que las salidas sean igual que las entradas de los módulos siguientes.

Cambios en los módulos

Se realizaron cambios en los siguientes módulos:

- Alu control
- Single Datapath
- Unidad de control

En la Alu control y la unidad de control se implementaron las instrucciones de tipo I. Y en el Single Datapath se generaron nuevas conexiones y algunos otros cambios de conexiones que fueron adaptados para esta fase.

Scripter

Primero se trabajo con la propuesta de un programa en cual realizaría el algoritmo de las N-Reinas, sin embargo, se decidió cambiar la propuesta a la escritura en el banco de registros con la tabla ASCII.
En el proceso de la creación de este programa nos encontramos con varias problemáticas la primera seria que aún no entendíamos la lógica que debíamos utilizar para que con direcciones pudiéramos mover la información a nuestro gusto. Durante los avances que fueron surgiendo tanto individualmente como en equipo se fue la inicialización y una acumulación de valores bajo un solo registro y poder llegar a un numero dado el cual podría traducirse según los valores de la tabla ASCII en una letra, en este caso eran ocho letras ya que la palabra que se registraría seria la pablara "talachar" ya que es una palabra bastante usada.
Se busco una inicialización en ceros, pero nos dimos cuenta que realmente no era necesario ya que no necesitábamos inicializar, sino que necesitábamos reescribir. Queriendo hacer algo un poco más eficaz se genero una lista con la palabra "talachar" esta se escribiría con sumas instantáneas en el punto donde debería ir. Revisando las clases y otras fuentes de información nos dimos cuenta de como era la estructura para hacer un bucle dentro de nuestra arquitectura de tipo mips sin embargo nos dimos cuenta de que este bucle no era como el que nosotros necesitábamos exactamente. A la hora de aprenderlo, notamos el uso de registros especiales para poder usar los contadores, limites e inicios. Prácticamente se inicializaban registros

para ir tomando en cuenta el avance que se iba teniendo. Sin embargo nos estancamos en la parte más importante, ya que tenemos un numero de 8 o 9 instrucciones las cuales son dedicadas a la escritura de valores que se pueden traducir en caracteres según la tabla ASCII sin embargo es una repetición de un código parecido solo que cambia una de las direcciones, tomando eso en cuenta no podíamos encontrar la forma adecuada para que valla avanzando a uno par únicamente tener un bucle y no hacer una escrito de cada una de las instrucciones una por una tomando en cuenta que en nuestra memoria de instrucciones solo podemos poner instrucciones de las cuales debemos llevar inicializadas algunas por el mismo proyecto.

Conclusiones Proyecto final

Conclusión De La Cruz Chavarría Christian Israel

Trabajar esta fasé dos como Administrador fue más complicado de lo que parecía, totalmente lo fue. Era necesario para mí estar en todos lados viendo las distintas problemáticas, y como resolverlas. hubieron muchos errores en los campos de lenguaje ensamblador y de código verilog, así que como anterior descriptor de hardware, tuve que adentrarme mucho en eso, tomando en cuenta que en la base y el fin de nuestro proyecto. Por dichas razones descuide un poco mi propio campo, además de que me asustaba tratarlo dado mi sentir acerca de mi conocimiento en ensamblador. En la segunda semana de trabajo específicamente, me plantee cómo hacerlo o ignorando ese miedo, y además pedí ayuda a mis compañeros, lo cual resulto en un avance significativo del proyecto, entendí de manera concisa lo que necesitaba y quería hacer con el script y aunque nos estancamos en la parte de los ciclos, siento que comprendí bien la lógica de lo planeado a hacer y del avance que tuvimos.

Una última cosa que quisiera agregar, es que hubo un gran número de reuniones para platicar todo lo que íbamos haciendo, y que espero haber dejado en mis compañeros la idea de eso mismo, de que somos un equipo y que sabiendo nuestras fortalezas, podemos ayudarnos, que se puede hacer una reunion en discord a la hora que sea necesario en el momento que sea necesario para entender lo que estamos haciendo.

Conclusión Ramírez Vázquez Ezequiel Alberto

Mi conclusión sobre esta fase 2 respecto al código en verilog es que se me hizo entretenido e interesante ver como se aplican las instrucciones de tipo I y también me gusto ver como se formaban sus buffers y los minis vectores que llevan los buffers ya que con esta parte no tuve mucha complicación.

Lo que no me gusto sobre esta fase en el código en verilog es hacer todas las instancias ya que son demasiados cables y me confundía con los cables ya creados antiguamente implementados en la fase 1 ya que tuve que ir viendo paso por paso los cables que ya estaban creados y me servían para esta fase y crear los nuevos cables para después volver a instanciar todos los módulos y que no hubiese un error en las instancias.

Conclusion Oscar Beltran Villegas

Con esto se concluyó el script de lenguaje ensamblador a binario de la parte del tipo de instrucción I, ya que seguimos con esa después de realizar con las instrucciones tipo R, el código se alargó un poco más de lo esperado pero los resultados fueron excelentes, se prevé que se mejorara en la parte de la interfaz y se tratara de hacer cada vez lo más compacto posible.

Con esto se concluyó el script de lenguaje ensamblador a binario de la parte del tipo de instrucción I, ya que seguimos con esa después de realizar con las instrucciones tipo R, el código se alargó un poco más de lo esperado pero los resultados fueron excelentes, se prevé que se mejorara en la parte de la interfaz y se tratara de hacer cada vez lo más compacto posible.

Conclusión Mariscal Viorato María Fernanda

Mi conclusión en esta fase en la documentación para mi fue un poco intenso, ya que estuve recabando mucha información y tratando de comprender y simplificar todo de tal manera que a la hora de trascribirlo pudiera verse lo mas claro posible o en su defecto lo menos revoltoso. Creo que en esta fase dos nos unimos mas como equipo por el grado de dificultad que esta tenía ya que ubo muchas mas reuniones, mas dudas y creo que todos tratamos de ayudar con todo.

Fue un poco complicado el tema del código en ensamblador sin embargo siento que fue divertido y pude aprender un poco mas tanto de los módulos como de mis compañeros.

Cuadros de referencias y practicas

Registers Bank			Memorie	
I	V		I	V
0	5	indice	0	0
1	4	limite sup	1	
2	4	limite inf	2	
3	0	valor	3	
4			4	
5	0		5	
6	0		6	
7	0		7	
8	0		8	
9	0		9	
10	0		10	

	Intructions				
0	tipo	rt	rs	in	
1	addi	$0	$0	#5	
2	addi	$1	$0	#20	
3					Inicializacion FOR
4	addi	$3	$0	#0	
5			BASE	OFF	
6	sw	$3	$3	#0	
7	addi	$2	$0	#4	escritura
8	addi	$1	$1	#-1	paso
9	beq	$1	$2	#1	condicion
10	j	#6			
11					
12	beq	$0	$1	#1	

Evidencias del código funcionando

REFERENCIAS

Computer Organisation and Design -David A. Patterson- Jhon L.Hennessy (2014).pdf

MIPS- Architecture for programmers- volume II-A: The MIPS32-Document- MD00086 (2016).pdf